Seminario – Seguridad en desarrollo del Software

Tema: Principios, identificación y riesgo

Autor: Leudis Sanjuan

Principios para una codificación segura

Minimizando el área de la superficie de ataque

Cada elemento que se añade a una aplicación añade una cierta cantidad de riesgo a la aplicación total. El objetivo del desarrollo seguro es reducir el total del riesgo reduciendo el área o escenario de ataque.

Por ejemplo, una aplicación Web implementa ayuda en línea con una función de búsqueda. La función de búsqueda puede ser vulnerable a ataques de inyección de SQL.

Una inyección SQL sucede cuando se inserta o "inyecta" un código SQL "invasor" dentro de otro código SQL para alterar su funcionamiento, y hacer que se ejecute maliciosamente el código "invasor" en la base de datos. Este tema será tratado en un módulo a seguir.

Ahora bien, continuando con el ejemplo, si la característica de ayuda se hubiera limitado a usuarios autorizados la probabilidad del ataque se hubiera reducido.

Pero si la característica de ayuda de la función de búsqueda fuera introducida a través de rutinas de validación de datos, la habilidad para realizar ataques de inyección SQL se hubiera reducido dramáticamente, inclusive si esta estuviera disponible para cualquier usuario de Internet.

Seguridad por defecto

Como desarrolladores debemos identificar las diferentes situaciones que pueden generar un riesgo e incluir funciones en nuestras aplicaciones que permitan reducirlo.

Por ejemplo, por defecto debe habilitarse una función que valide la complejidad de la definición de una contraseña y su tamaño. En caso de no ser así, los usuarios pueden definir contraseñas fáciles de deducir y por tanto inseguras.

Principio del mínimo privilegio

Este principio significa que los usuarios de un sistema sólo deben tener habilitado, exclusivamente, los derechos de acceso (escritura, lectura, etc.) a los objetos que ineludiblemente requieran para cumplir las funciones del puesto que ocupan.

Por ejemplo, si un usuario desde su equipo debe tener acceso de lectura y escritura sobre una carpeta en un servidor, sólo se le deben otorgar los permisos de lectura y escritura y no del control total (administrador).

Revisar las transacciones

Revise especialmente las operaciones que se realizan con las bases de datos o con otros elementos críticos. Mida el riesgo y tome las acciones correctivas necesarias. Observe el siguiente código:

```
 isAdmin = true;
try {
 codeWhichMayFail();
    isAdmin = isUserInRole( “Administrator” );
}
   catch (Exception ex) {
   log.write(ex.toString());
}
```

Sí el código codeWhichMayFail() falla por alguna razón, el usuario es administrador por defecto. Obviamente esto es un riesgo de seguridad.

Utilice las herramientas de análisis de código

En el mercado existen muchas herramientas que pueden analizar el código y detectar errores de seguridad. Estas herramientas encuentran errores ocultos con mayor eficacia y menos esfuerzo. Por ejemplo: para el caso de C y C+ existe *Visual Studio Team Edition* para *Developers*.

Realice una revisión de seguridad completa

El objetivo de toda revisión de seguridad es mejorar la seguridad de los productos que ya se han lanzado (mediante revisiones y correcciones) o garantizar que no se distribuya ningún producto nuevo hasta que sea lo más seguro posible.

No revise el código de forma aleatoria. Prepare de antemano la revisión de seguridad y comience por crear un modelo exhaustivo de amenazas. Dé prioridad

al código que debe recibir la revisión de seguridad más intensa e indique qué errores de seguridad hay que buscar.

Sea concreto en cuanto a lo que se debe buscar durante una revisión de seguridad. Cuando se buscan problemas concretos, normalmente se encuentran. Examínelo con mayor atención si usted encuentra varios errores de seguridad en una sección; probablemente indique que existe un problema de arquitectura que hay que solucionar. Si no encuentra ningún error de seguridad, normalmente significa que la revisión de seguridad no se realizó correctamente.

Realice revisiones de seguridad como parte de la estabilización de cada hito y de la inserción de líneas de productos mayores establecidas por la dirección.

Utilice una lista de comprobación de revisión de código para mayor seguridad

Independientemente de la función en el desarrollo de software, resulta de gran utilidad disponer de una lista de comprobación que seguir, a fin de garantizar que el diseño y el código cumplen con los requisitos mínimos.

Valide todos los datos introducidos por los usuarios

Si permite que la aplicación acepte datos proporcionados por los usuarios, directa o indirectamente, debe validar esos datos antes de utilizarlos. Los usuarios malintencionados intentarán provocar errores en la aplicación manipulando los datos de modo que representen datos no válidos. La primera regla para los datos introducidos por el usuario es: todos los datos son erróneos hasta que se demuestre lo contrario.

Extreme las precauciones cuando utilice expresiones regulares para validar los datos introducidos por el usuario. Para expresiones complejas como las direcciones de correo electrónico, es fácil pensar que se efectúa una validación completa cuando no es así. Pida a sus compañeros que revisen todas las expresiones regulares.

Valide perfectamente todos los parámetros de las interfaces de programación de aplicaciones (API) exportadas

Asegúrese de que todos los parámetros de las API exportadas son válidos. Esto incluye los datos que parecen ser coherentes pero que están más allá del intervalo de valores aceptado, como los tamaños de búfer excesivos. No utilice aserciones para comprobar los parámetros de las API exportadas, porque las aserciones se quitarán en la versión de lanzamiento.

Utilice las API criptográficas

En lugar de escribir a su propio software criptográfico, utilice API criptográfica desarrolladas por firmas reconocidas (IBM, Microsoft, etc.). Al utilizar API criptográfica, los desarrolladores quedan libres para concentrarse en la generación de aplicaciones.

Algunas recomendaciones adicionales

- Tenga en cuenta los errores de seguridad se pueden replicar

  Esto dado que con frecuencia en vez de escribir código los reutilizamos basados en un código anterior.

  Por eso, cuando nos encontremos con una funcionalidad que debemos implementar en varias aplicaciones, es preferible crear una función, una rutina, un servicio o un componente que sea reutilizable para cualquier aplicación. Por ejemplo: envío de correo, autenticación de un sistema, validación de un formulario.

  De esta forma, si encontramos un error en ese tipo de funcionalidades, la falla es más fácil de corregir.

- Revise todas las rutas de acceso a errores

  A menudo, el código de las rutas de errores no se prueba bien y no limpia todos los objetos, como los bloqueos o la memoria asignada. Revise con atención estas rutas de errores y, si es preciso, cree pruebas de inserción de errores para ejecutar el código.

- Privilegios de administrador para que se ejecute la aplicación

  Las aplicaciones deben ejecutarse con el menor privilegio necesario para realizar el trabajo. Si un usuario detecta una vulnerabilidad de seguridad e inserta un código en el proceso, este código malintencionado se ejecutará con los mismos privilegios que el proceso huésped. Si el proceso se ejecuta como administrador, el código malintencionado se ejecuta como administrador.

- Rutas de acceso a archivos y direcciones URL canónicas

  Evite situaciones donde sea importante la ubicación de un archivo o una dirección URL. Utilice ACL del sistema de archivos en lugar de reglas basadas en nombres de archivo canónicos.

- Evite las saturaciones del búfer

  Una saturación del búfer estática se produce cuando un búfer declarado en la pila se sobrescribe porque se copian datos más grandes que él. Las variables declaradas en la pila se ubican junto a la dirección de devolución del llamador de la función. Las saturaciones del búfer son bastantes peligrosas.

  Normalmente, la causa se debe a datos introducidos por el usuario que se pasan a una función; por ejemplo, para el caso de C, la función strcpy(), con el resultado de que la dirección de devolución de la función se sobrescribe por una dirección elegida por el atacante. Para evitar las saturaciones del búfer es fundamental escribir una aplicación robusta.

- Combinaciones de Id de usuario y contraseña dentro del código

  No utilice contraseñas contenidas dentro del código. Modifique el instalador para que, al crear las cuentas de usuario integradas, se soliciten al administrador contraseñas seguras para cada cuenta. De esta manera, se puede mantener la seguridad de los sistemas de nivel de producción del cliente.

Seminario (seguridad en desarrollo del software)

Bibliografía

Choo, C.Information Management for the Intelligent Organization, Third Edition. Published by Information Today/Learned Information

http://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents

http://www.wikilearning.com/tutorial/como_escribir_programas_seguros-como_escribir_programas_seguros/6795-1

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/cryptography_reference.asp

http://msdn.microsoft.com/es-es/library/ms182023(VS.80).aspx